NASA
Technical Memorandum 100889

AVSCOM
Technical Report 88-C-011

A Microprocessor-Based Real-Time Simulator of a Turbofan Engine

Jonathan S. Litt
Propulsion Directorate
U.S. Army Aviation Research and Technology Activity—AVSCOM
Lewis Research Center
Cleveland, Ohio

and

John C. DeLaat and Walter C. Merrill
Lewis Research Center
Cleveland, Ohio

Prepared for the
19th Annual Pittsburgh Conference on Modeling and Simulation
cosponsored by the ISA and IEEE
Pittsburgh, Pennsylvania, May 5–6, 1988

NASA

US ARMY
AVIATION
SYSTEMS COMMAND
AVIATION R&T ACTIVITY

A MICROPROCESSOR-BASED REAL-TIME SIMULATOR OF A TURBOFAN ENGINE

Jonathan S. Litt
Propulsion Directorate
U.S. Army Aviation Research & Technology Activity - AVSCOM
Lewis Research Center
Cleveland, Ohio 44135

John C. DeLaat and Walter C. Merrill
National Aeronautics and Space Administration
NASA Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

A real-time digital simulator of a Pratt and Whitney F100 engine is discussed. This self-contained unit can operate in an open-loop stand-alone mode or as part of a closed-loop control system. It can also be used in control system design and development. It accepts five analog control inputs and its sixteen outputs are returned as analog signals.

INTRODUCTION

Background

The simulator [1] is based upon a HYTESS-like model [2 and 3] of the F100 engine without augmentation (without afterburning). HYTESS is a simplified simulation written in FORTRAN of a generalized turbofan engine. To create the simulator, the original HYTESS program was revised to incorporate F100 specific parameters. Additionally, other code was adapted from the Advanced Detection Isolation and Accommodation (ADIA) program [4] running in the Control Interface and Monitoring (CIM) Unit [5].

The F100 engine is a high performance, twin-spool, low by-pass ratio, turbofan engine. Figure 1 shows the locations of the engine inputs defined in Appendix C. Figure 2 shows the locations of the engine sensors defined in Appendix D.

Purpose

The F100 engine simulator was designed to support the ADIA F100 engine test. The ADIA engine test was the culmination of a research project aimed at showing that, using a computer model of the engine, the control system can continue to control an engine (even during transients) with one or more of the engine sensors giving false readings. The objective of this engine test was also to demonstrate that the ADIA software works on a real engine and is, therefore, reliable and useful in a real environment. This software had already been successfully tested on a Hybrid Computer Simulation [6]. Due to anticipated uncertainties in the set-up in the test cell, it was determined well in advance of the test run that changes to the CIM Unit's software would be necessary. To facilitate these changes the simulator was connected in parallel with the real engine in the Propulsion Systems Laboratory (PSL) as shown in figure 3. The simulator is a portable box which could be taken into PSL to verify any changes in the CIM Unit's software before they were tried out on the engine. This technique prevents damage to the system being controlled which might otherwise occur if the controller's software contains a serious error.

Order of the report

This report will discuss the simplified engine model which was used. It will also briefly go over the actuator and sensor models employed. It will describe the actual implementation including some hardware issues and discuss the individual subroutines used. A user's manual is included with step by step instructions of how to use the system. Finally performance comparisons with the real engine will be presented.

MODEL

The original full nonlinear model for the F100 engine is a 13 000 line FORTRAN program. This model accurately reproduces the true engine dynamics over the full flight envelope but is so numerically intensive that it cannot be run in real-time on a standard computer.

Since the main objective of the simulator was that it had to run in real-time, a HYTESS-like design was developed. The HYTESS model is a much more efficient representation of the engine dynamics than the full nonlinear model but the penalty for this is that the relationship between physical elements of the engine is lost.

The HYTESS model is set up in state space form using the vector differential equations

$$\begin{aligned} \dot{x} &= f(x,u,\Phi) \\ y &= g(x,u,\Phi) \end{aligned} \tag{1}$$

where x is the vector of intermediate engine variables or states, $\dot{x}$ is the derivative of x with respect to time, u is the vector of control inputs, Φ is the vector of environmental conditions, and y is the vector of engine outputs. Clearly, at steady-state points,

$$\begin{aligned} \dot{x} &= f(x_b,u_b,\Phi_b) = 0 \\ y_b &= g(x_b,u_b,\Phi_b) \end{aligned} \tag{2}$$

where the subscript b denotes a steady-state point on the operating line known as a base point. In other words, selecting y_b and Φ_b vectors determines steady-state x_b and u_b vectors such that the quadruple (x_b,y_b,u_b,Φ_b) satisfies (2). The base points from the HYTESS model representative of the entire flight envelope are shown in figure 4.

Generally, state-space equations of a system linearized about the operating point (x_b,u_b,y_b) are of the form

$$\begin{aligned} \dot{x} &= F(x - x_b) + G(u - u_b) \\ y &= y_b + H(x - x_b) + D(u - u_b) \end{aligned} \tag{3}$$

where F, G, H, and D are matrices of appropriate dimension. The F100 model was linearized at each base point using perturbation techniques. Thus, the state-space model is accurate in the neighborhood of a base point and the model behaves in a similar manner to a linear system about the base point. The actual equations used in the model are of the form

$$\begin{aligned} \dot{x} &= F(y,\Phi)[x - x_{ss}] \\ y &= y_b(y,\Phi) + H(y,\Phi)[x - x_b(y,\Phi)] + D(y,\Phi)[u - u_b(y,\Phi)] \end{aligned} \tag{4}$$

where x_{ss} is given by

$$x_{ss} = x_b(y,\Phi) - F^{-1}G(y,\Phi)[u - u_b(y,\Phi)]$$

where the subscript ss denotes a steady-state point near a base point. It is clear that the Equations for y in (3) and (4) are equivalent. To show that the equations for $\dot{x}$ are also the same, the equation for x_{ss} must be substituted into the equation for $\dot{x}$ in (4) as follows:

$$\begin{aligned} \dot{x} &= F(y,\Phi)[x - x_{ss}] \\ &= F(y,\Phi)[x - \{x_b(y,\Phi) - F^{-1}G(y,\Phi)[u - u_b(y,\Phi)]\}] \\ &= F(y,\Phi)x - F(y,\Phi)\{x_b(y,\Phi) - F^{-1}G(y,\Phi)[u - u_b(y,\Phi)]\} \\ &= F(y,\Phi)x - F(y,\Phi)x_b(y,\Phi) + FF^{-1}G(y,\Phi)[u - u_b(y,\Phi)] \\ &= F(y,\Phi)[x - x_b(y,\Phi)] + G(y,\Phi)[u - u_b(y,\Phi)] \end{aligned}$$

Therefore the systems of Equations in (3) and (4) are equivalent.

In the F100 model as in HYTESS, the elements of the matrices F, G, H, and D are nonlinear polynomials. These polynomials were determined by a curve-fitting algorithm used to regress each matrix element upon elements of y and Φ or upon elementary functions of y and Φ. Thus the polynomial matrices approximate the data points, i.e., they approximate the system matrices determined using perturbational techniques at each base point. Therefore, at each point in the envelope, the polynomials need only be evaluated to determine the system matrices.

The actuators and sensors are, for the most part, modeled as first-order lags with a small dead zone or other small nonlinearity included. The time constants used are similar to those used on the hybrid simulation and are very close to those of the real instrumentation being modeled.

IMPLEMENTATION

The simulator itself fits into a single rack-mountable Zendex ZX-660(A) chassis. This chassis contains nine Multibus/IEEE 796 compatible expansion slots and power supply. In addition, an Intel MDS 730 rack-mountable dual 8 in. floppy disk drive unit and a terminal device are required to run the program. The chassis contains the five boards shown in figure 5. The single board computer on which the simulation runs is an INTEL 86/30 with an 8086 chip, an 8087 floating point coprocessor, and 256Kb of memory. (This is an expansion from the original 64Kb and is required to load the FORTRAN code even though the operating system limits the program size to not more than 64Kb.) A Zendex ZX-200A single board disk controller is included to communicate with the disk drives. A Data Translation DT 1742-32 DI is the third board. It contains 32 differential input channels, a multiplexer, and an A/D converter. Its purpose is to accept the analog control signals from the CIM Unit and digitize them. There are also two Data Translation DT 1842-8-V 8 channel D/A boards which convert all of the simulated outputs to analog form for output.

The software consists of 21 routines - 11 in FORTRAN and 10 in 8086 and 8087 assembler. In addition there are four libraries required by the program.

There are several modes in which the simulation can run,depending upon the application. They are: initialization/run, PSL/hybrid, calibration, open-loop/closed-loop, and actuator (Appendix B).

Description of Modes

Initialization/run

The initialization/run mode is a consequence of the fact that the simulation is not fast enough to accurately model the whole flight envelope dynamically in real time. The ADIA control interval was 40 ms. It was determined that for proper stability and accuracy a good rule of thumb is that a numerical (Euler) integration time of not more than one quarter the control interval should be used in the simulation. This constraint came about from a desire to reduce the interaction between the simulation and the control by reducing any phase shift due to time delays in the simulation as much as possible. As a full envelope simulation, the minimum achievable update time (integration time) was approximately 40 ms or four times the desired interval. To overcome this problem, a drastic reduction in the cycle time of the algorithm was required. It was possible to determine the length of time each subroutine took to run. The FORTRAN code had already been optimized [7] so the length of time each routine took was essentially the minimum possible. Therefore, short of putting the simulation on multiple computers (parallel processing) or using a faster processor which was not feasible at the time, the most reasonable solution to the time problem was to change the simulation to a steady-state model. This consisted of calculating the base points and the matrix elements in non-real time (these were the longest routines) and then, in the real time loop, evolving the system as a linear system to the new operating condition. The result is a linear model valid within a small region about a given operating point. This model gives excellent steady-state results and good transient results for small perturbations, such as small movements of the Power Lever Angle (PLA). However, the model

will not perform accurately for large perturbations such as large PLA movements.

Propulsion Systems Lab Mode (PSL)

The next mode is PSL mode. It allows the scaling of the control signals and the simulator outputs to correspond to those of the engine in PSL. Initially the inputs and outputs of the simulator were scaled identically to the inputs and outputs of the F100 Hybrid Simulation. These were all ±10 V, straight line representations of the engine inputs and outputs. However, the actual engine inputs and outputs consist of linear pots, resolvers, thermocouples, flowmeters, and electro-hydraulic actuators. These devices typically do not accept or produce ±10 V, linear signals. Thus, while the system was in PSL, the scaling for the control inputs from the CIM Unit to the engine simulator had to be mapped to the equivalent scaling for the hybrid simulation. Likewise, the scaling for the outputs of the simulator's sensors had to be mapped to the equivalent scaling values for the actual engine sensors so the CIM Unit received the same values the engine sensors would produce.

Calibration

The calibration mode is used to test the mappings. Once a map has been determined and implemented, it must be tried out with the simulation. The calibration switch allows the user to bypass the system evolution subroutines so he can set an intermediate value and examine the corresponding value which is used as output. In the same way, the simulator can receive analog input values and the user can examine the commanded values once they have gone through the conversion. Using these two methods, the user can tell if the values are being mapped correctly from the PSL values to the hybrid values and vice versa.

Closed loop/open loop

There is also a closed-loop and corresponding open-loop mode which allow the simulator to receive the control signals from either an outside source such as the CIM Unit or from stored in its own memory, respectively.

Actuator

The last switch is used to simulate only the actuator models. To ensure that the real actuators are all working correctly and since they are quite simple to model accurately, the simulator can be run in parallel with the engine and the actuator feedback values compared. The only difference between this and the standard run loop of the simulation is that TT2, the only independent variable which the actuators require beside the control signals, is read in from the CIM Unit rather than calculated. Since no other information is required and the actuator calculations are fairly simple, this can be used as a full envelope real-time simulator for the actuators. Of course the engine model outputs are meaningless in this mode because the base points are not being calculated.

The modes are all set by software switches which can be toggled using MINDS [8]. MINDS is a program used to examine and to set values of memory locations. To the user, MINDS looks like an interpreter. The user types in commands and MINDS carries them out. MINDS runs in the background; it is interrupted by every other program but even though it runs for only about 17 percent of the time in the run-time loop, to the user it seems as if it is running continuously. The user just types in commands and MINDS picks them up as the program cycles. It carries out the commands and returns the response and the MINDS prompt almost immediately.

The system runs under CP/M V2.2 and has a limitation that the total space for code and data be not more than 64Kb. With a reduced capability version of MINDS included, the total memory required for the program is about 50Kb, approximately two-thirds of which is code and one-third is data.

OPERATING PROCEDURE

After the system is booted, the program can be run by typing the name of the disk drive where the program disk is located followed by a colon and the

name of the program. When the RETURN key is pressed, the executable code is brought in from disk and run.

The program starts running in the executive (figure 6). It initializes the update intervals, sets up the memory appropriately and takes care of the administrative details. Then it executes a subroutine which initializes all of the constants such as time constants and calculates the exponents associated with each one. Once through this section, the program never returns to it, the assumption being that the set up information will never change. Then the program enters the initialization loop by setting the interrupt timer (figure 7). This loop is not running in real-time (it has no time dependency) but it repeats every 50 ms. The purpose of this loop is to calculate the base points for the operating point which is entered using MINDS. These base points are also stored as the set points for that operating condition in the open loop mode. The initialization loop consists of the inlet routine and the routine which calculates the system matrices. The program is ready to be used interactively once the MINDS prompt (›) comes up.

Setting the appropriate software switch puts the program into the real-time mode. Now the numerical integration occurs which brings the simulation from its previous steady-state point up to the new steady-state point with a linear, non-realistic transient. The new steady-state point is, however, accurate and realistic. This loop has an update interval of 12 ms and during that time the control input routine, actuator routine, the system evolution routine (numerical integration), and the output signal routine all run. Any spare time is used up by the message generation routine or MINDS. The message generation routine takes priority over MINDS if it needs to run but it is only used to print out error messages. A more in-depth description of the simulator's operation is given in Appendix A.

Major Subroutines

At the beginning two routines are executed, each once only per run. They are called MSET and MTRXST and are simply routines for initialization of constants. After these are run, the program goes into the initialization loop. Here it executes INLET which calculates the ambient conditions based on the altitude and Mach number. Then it goes to EMODEL which determines the matrix elements by evaluating polynomials whose coefficients are functions of the ambient conditions. The scheduled values of engine variables are calculated in the subroutines RPFAND and RPLIMD which are called from EMODEL. Any extra time in this loop is used by MINDS to accept inputs from the user. He can change altitude and Mach number and the next time through the loop, everything is recalculated for the new conditions. Since everything in the initialization loop is calculated directly, the loop need only be executed once after a change is made for the values to be correct. The user can also set the switch to go from the initialization loop to the run loop while in MINDS. The update interval is short enough to essentially guarantee that the loop will be executed at least once after the conditions are changed to obtain the correct values before the switch can be set.

The run loop consists of the dynamic routines. The first section reads in the control signals from the CIM Unit and converts the scaled integers to real numbers. ACTUAT takes the real commanded values and evolves the actuator models to their value at the current time step. This output is used by EVOLVE to integrate the differential equations describing the engine itself. The engine outputs, actuator feedbacks, PLA, and the ambient conditions are then converted to scaled integers and sent via D/A converters to the CIM Unit. The I/O sections are part of the multiplexer interrupt service routine section of the executive.

Many of the routines listed above call their own subroutines which do table look-ups or some type of calculation. The relationships are shown in figure 8.

Error Handling

Most types of errors that occur produce an interrupt and are handled by interrupt service routines. In general, one of the results of these routines is to give the user an indication that the error took place. If a non-catastrophic error occurs the interrupt service routine signals a message to be printed. This printing is done in the remaining time at the end of the

run loop. Printing out a message is a slow process and may take several cycles of the run loop to complete. Because more than one error might occur in one cycle and each takes so long to print, a data structure is used to store the starting addresses of each error's corresponding message. Up to 15 addresses can be held in this circular queue.

At the end of the simulation time in the run loop, the program checks if the queue is empty and, if not, whether there is a message already being printed. If no printing is in progress and a message is waiting to be printed, the program will initiate printing the one at the head of the queue. In all other cases it returns to what it was doing before the interrupt came in which started the current cycle of the run loop - either MINDS or printing a message.

SIMULATION RESULTS

The steady-state accuracy of the model is excellent. This is because the HYTESS model was based on the steady-state performance of a turbofan engine and the base point calculations which define steady state performance in HYTESS were derived from steady-state data. The actuator only simulation is highly accurate in the real-time loop, both in steady-state and transient behavior. The full engine transient performance for small perturbations about a given operating point is also quite good. The full engine large perturbation transient performance leaves much to be desired since the engine model behaves like a linear system in the run loop.

CONCLUSIONS

Tests conducted in conjunction with the F100 Hybrid Simulation evaluation of the ADIA algorithm showed that the simulator works well as a real-time, steady-state and small perturbation substitute for the full Hybrid, nonlinear simulation. The full-scale engine demonstration of the ADIA proved the capabilities of the simulator as a real-time code verifier and as a full envelope, real-time actuator simulator for actuator fault detection. This real-time, portable simulator capability will be valuable in future engine tests. With the rapid increases in microprocessor capabilities that have occurred since the F100 simulator was built, it is conceivable that full envelope, full engine simulation can now be achieved in real-time.

References

1. J.S. Litt, J.C. DeLaat, and W.C. Merrill, "A Real-time Simulator of a Turbofan Engine," NASA TM-100869, 1988.
2. W.C. Merrill, et. al., "HYTESS - A Hypothetical Turbofan Engine Simplified Simulation," NASA TM-83561, 1984.
3. W.C. Merrill, "HYTESS II - A Hypothetical Turbofan Engine Simplified Simulation With Multivariable Control and Sensor Analytical Redundancy," NASA TM-87344, 1986.
4. J.C. DeLaat and W.C. Merrill, "A Real-Time Implementation of an Advanced Sensor Failure Detection, Isolation, and Accommodation Algorithm," NASA TM-83553, 1984.
5. J.C. DeLaat and J.F. Soeder, "Design of a Microprocessor-Based Control, Interface and Monitoring (CIM) Unit for Turbine Engine Controls Research," NASA TM-83433, 1983.
6. W.C. Merrill, J.C. DeLaat, and W.M. Bruton, "Advanced Detection, Isolation, and Accommodation of Sensor Failures - Real-time Evaluation," NASA TP-2740, 1987.
7. J.C. DeLaat, "A Real-time FORTRAN Implementation of a Sensor Failure Detection, Isolation and Accommodation Algorithm," Proceedings of the 1984 American Control Conference, San Diego, CA, June 6-8, 1984.
8. J.F. Soeder, "MINDS - A Microcomputer Interactive Data System for 8086-Based Controllers," NASA TP-2378, 1985.

APPENDIX A

User's Manual for F100 Engine Simulator

1. Turn on all of the equipment, i.e. the chassis, the disk drive, and the terminal.
2. Insert the system disk into drive a: and the program disk into drive b:.
3. Boot the system by pressing the RESET button on the chassis.

4. When the system has booted, load and start the program by typing b:<program-name><RETURN>.
5. This causes the program to start executing. It goes through the one-time initialization routines, MSET and MTRXST, and enters the initialization loop containing INLET and EMODEL. In the spare time in this loop, MINDS runs, allowing the values of variables and flags to be changed. The MINDS variable definitions must either be entered by hand or loaded from a disk. Choose the mode in which the program is to be run. This can be changed at any time very simply. The default mode is initialization/hybrid/open-loop. Each switch (flag) can be changed independently.
6. Altitude and Mach number, ALT and XM0 respectively, can be changed through MINDS. The ambient conditions, which are all calculated in INLET, depend on them. For changes of these two variables to have any effect, the program must go through the initialization loop one time. The base points are calculated here and their values are stored for the additional purpose of being the set points in the open-loop mode.
7. Setting the value of RLOOP to 1 puts the simulation into the real-time run loop. The routines take about 10 ms to run leaving approximately 2 ms for MINDS provided there are no error messages to be printed. In this mode, MINDS can be used to check the value of variables and to switch modes.
8. Setting RLOOP to 0 again returns the program to the initialization loop but leaves the value of every variable unchanged. Thus a transient can be stopped and restarted (if the program is in open-loop mode) or the ambient conditions can be altered to move the system to another operating point.
9. To stop the simulation, reboot the system by pressing the RESET button on the chassis with the system disk in drive a:.

APPENDIX B

Software Switch	Comments
RLOOP	= 0, (default) program runs in initialization loop = 1, program runs in real-time run loop
PSL	= 0, (default) scaling of inputs and outputs corresponds to that of Hybrid simulation = 1, scaling of inputs and outputs corresponds to that of the Propulsion Systems Laboratory hardware
CALIB	= 0, (default) each routine in run loop is executed fully = 1, only the A/D converter and D/A converter routines are executed in the run loop, ACTUAT and EVOLVE are not. Thus the effect of scale factors for both input and output can be checked directly using MINDS
CLLOOP	= 0, (default) program runs in open-loop mode, command signals are taken from memory (the values can be changed using MINDS) = 1, program runs in closed-loop mode, analog command signals are read in through A/D converters
ACTSIM	= 0, (default) scheduled AJ (nozzle area) is proportional to the steady-state scheduled value calculated in RPLIMD = 1, scheduled AJ is calculated as a function of TT2 read in by the simulation at each control interval. This should only be used in the actuator simulation mode.

APPENDIX C

Input Channel	Variable	Comment
8	WFCOM	commanded main combustor fuel flow
9	AJCOM	commanded exhaust nozzle area
10	CIVVCM	commanded fan inlet variable vane angle
11	RCVVCM	commanded rear compressor variable vane angle
12	BLCCM	commanded compressor bleed (bleed is used open-loop)
13	TT2ACT	fan inlet temperature (used only in actuator mode)

APPENDIX D

Output Channel #	Variable	Comment
1	Timing DAC	
2	WFFBS	sensed main combustor fuel flow
3	AJS	sensed exhaust nozzle area
4	CIVVS	sensed fan inlet variable vane angle
5	RCVVS	sensed rear compressor variable vane angle
6	BLEBS	sensed compressor bleed (not used)
7	POS	ambient (static) pressure
8	PT2	fan inlet (total) pressure
9	TT2	fan inlet temperature
10	TT25	compressor inlet temperature
11	N1	sensed fan speed
12	N2	sensed compressor speed
13	PT4	sensed combustor pressure
14	PT6	sensed exhaust nozzle pressure
15	FTIT	sensed fan turbine inlet temperature
16	PLA	power lever angle

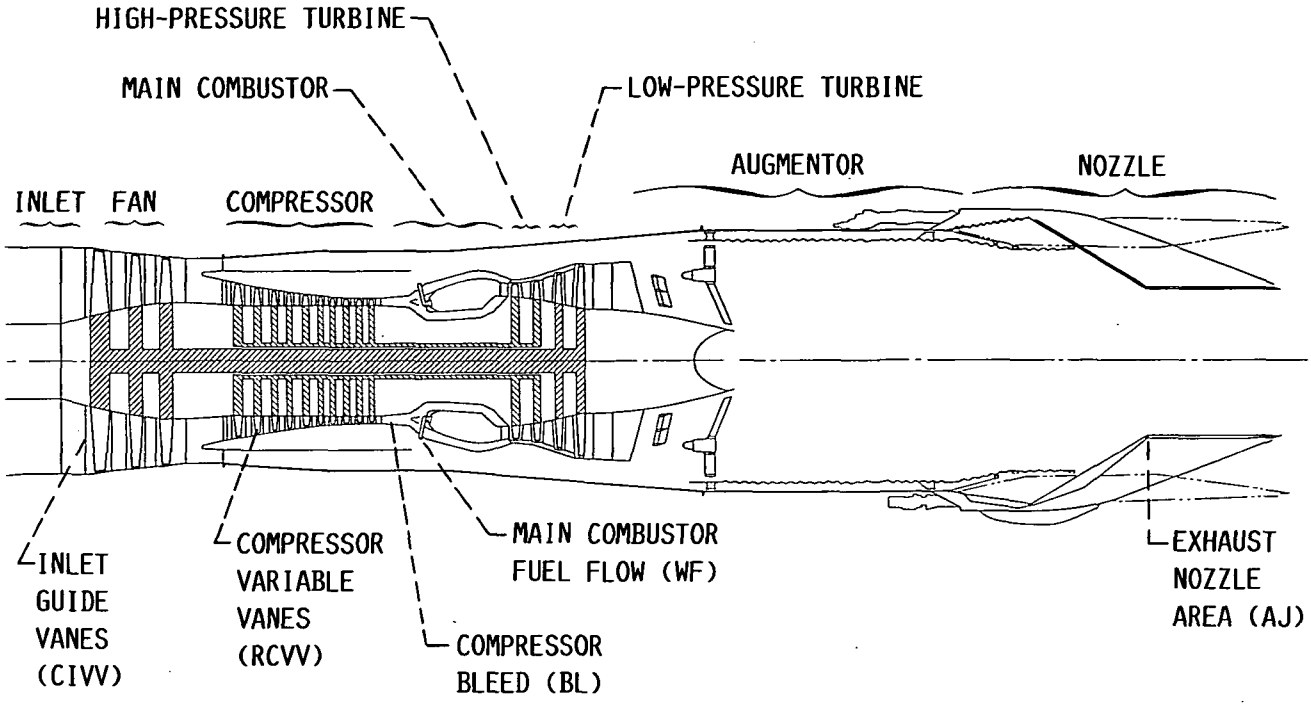

FIGURE 1. - F100 ENGINE INPUTS.

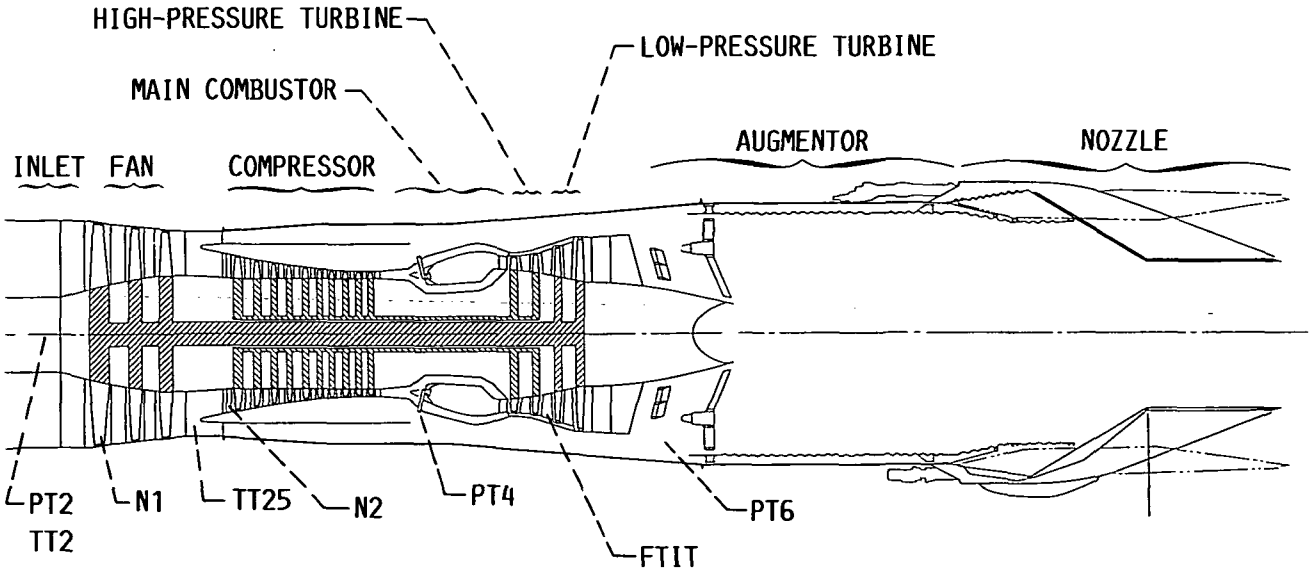

FIGURE 2. - F100 SENSE POINTS.

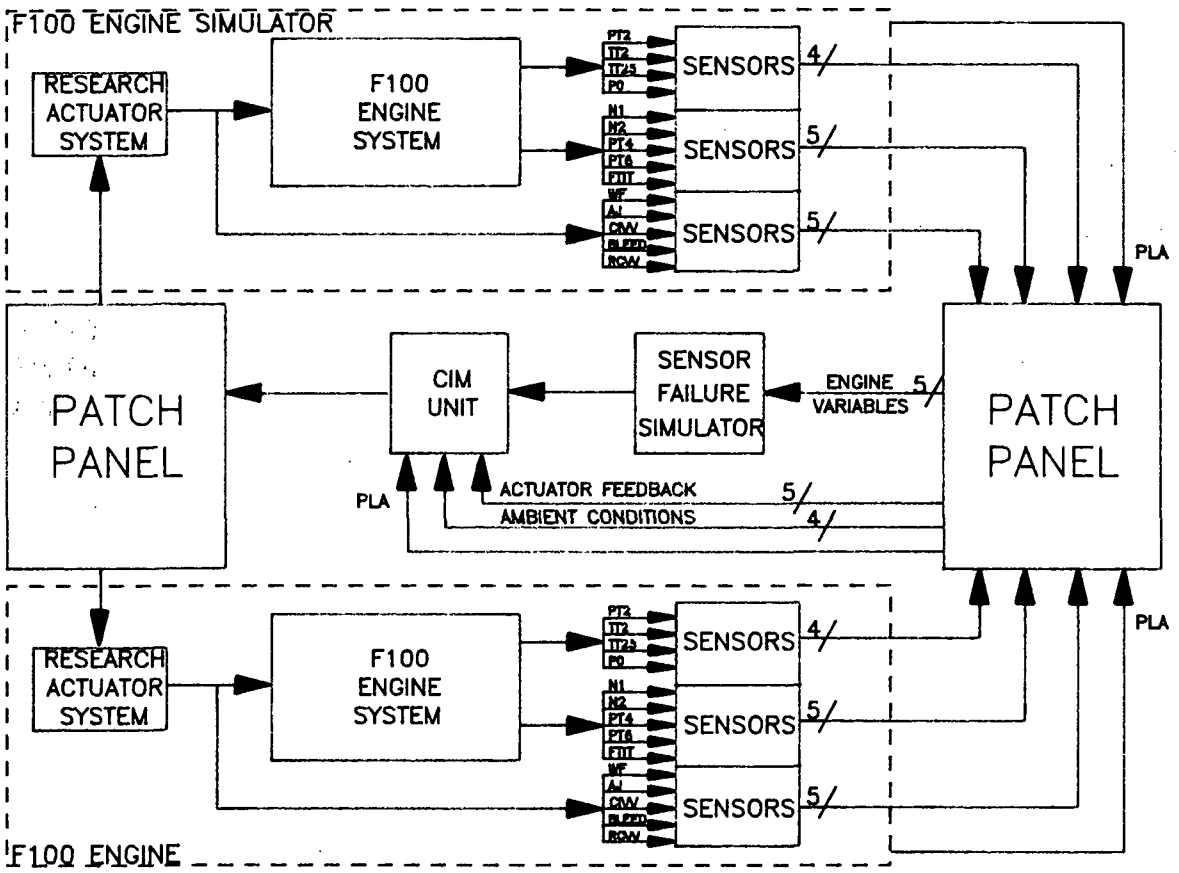

FIGURE 3. - TEST SETUP IN PSL.

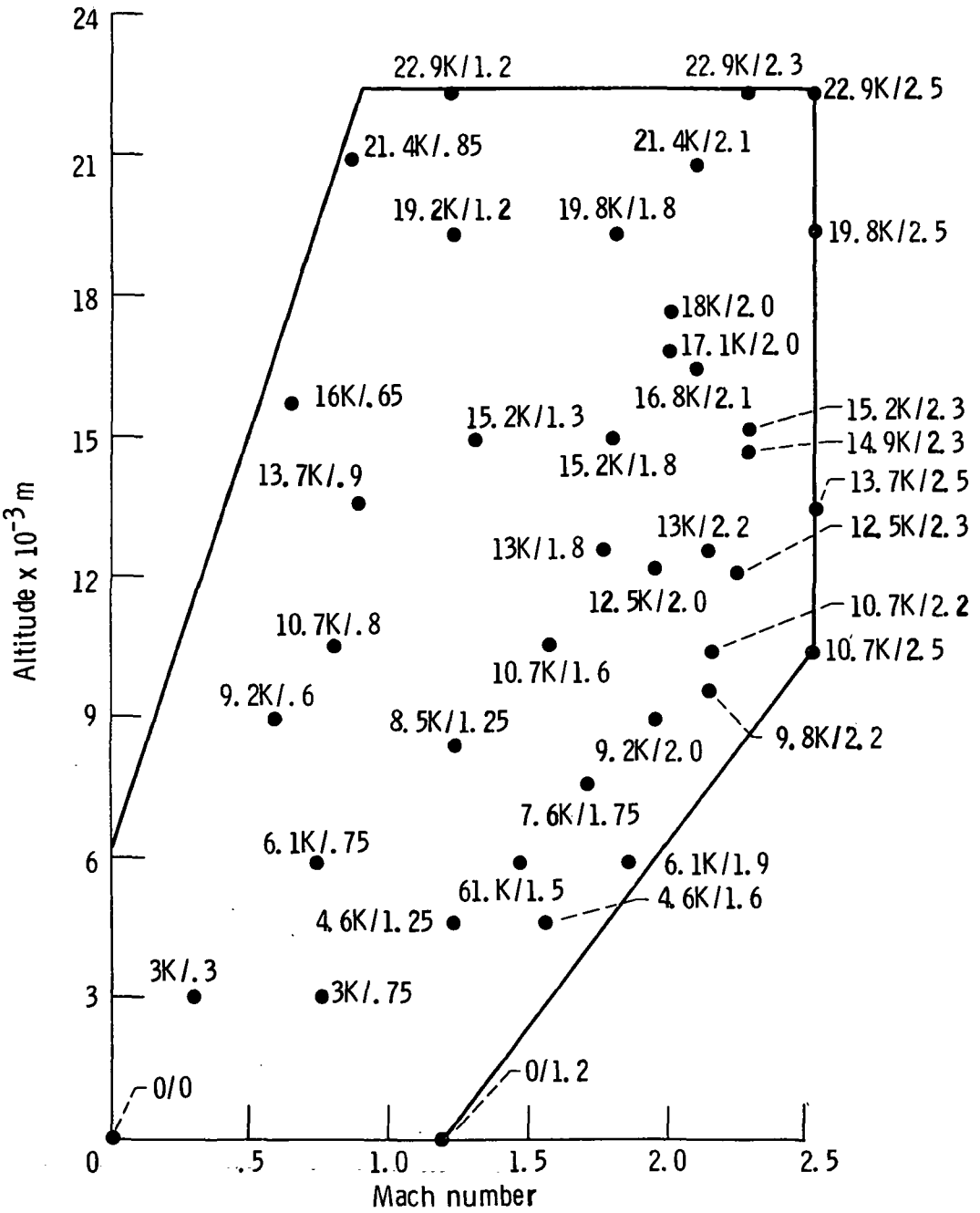

FIGURE 4. - FLIGHT ENVELOPE FOR HYTESS MODEL.

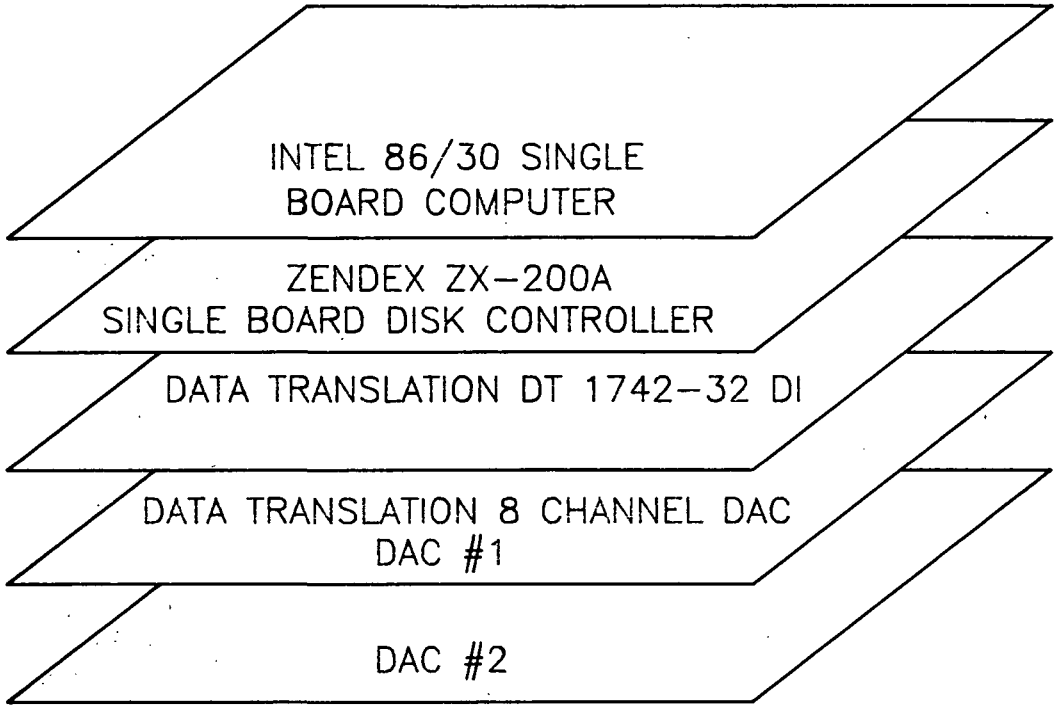

FIGURE 5. - ENGINE SIMULATOR HARDWARE.

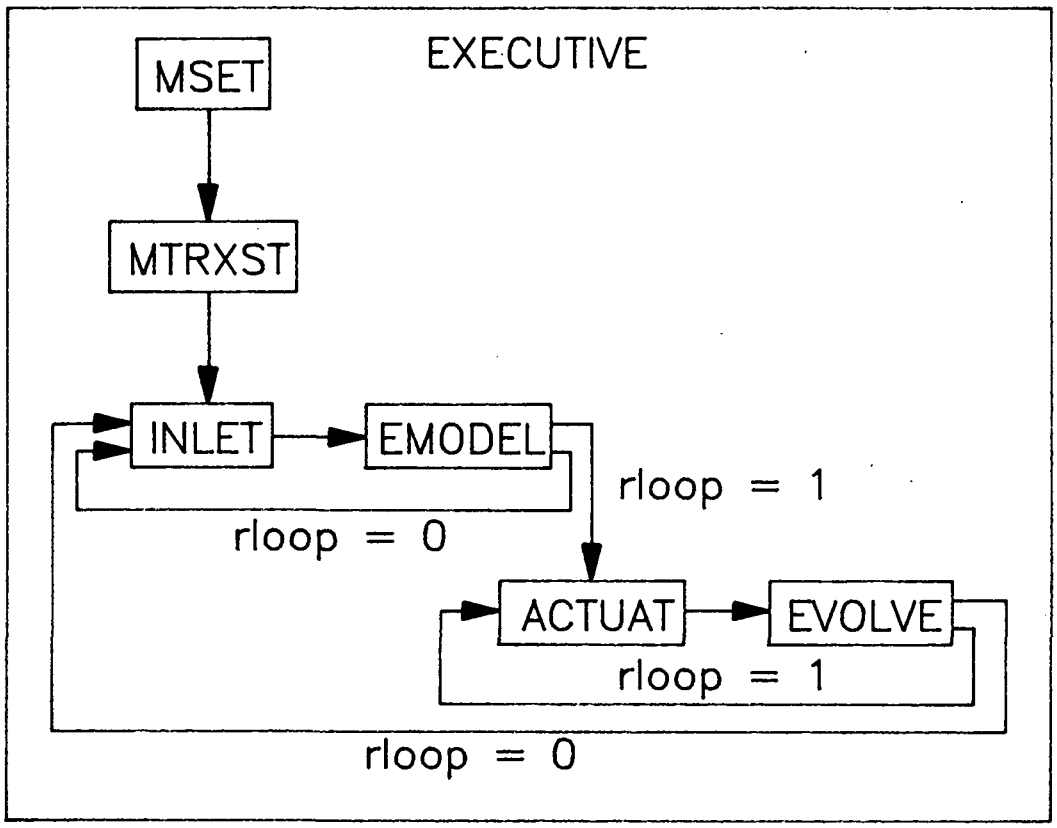

FIGURE 6. - PROGRAM FLOW.

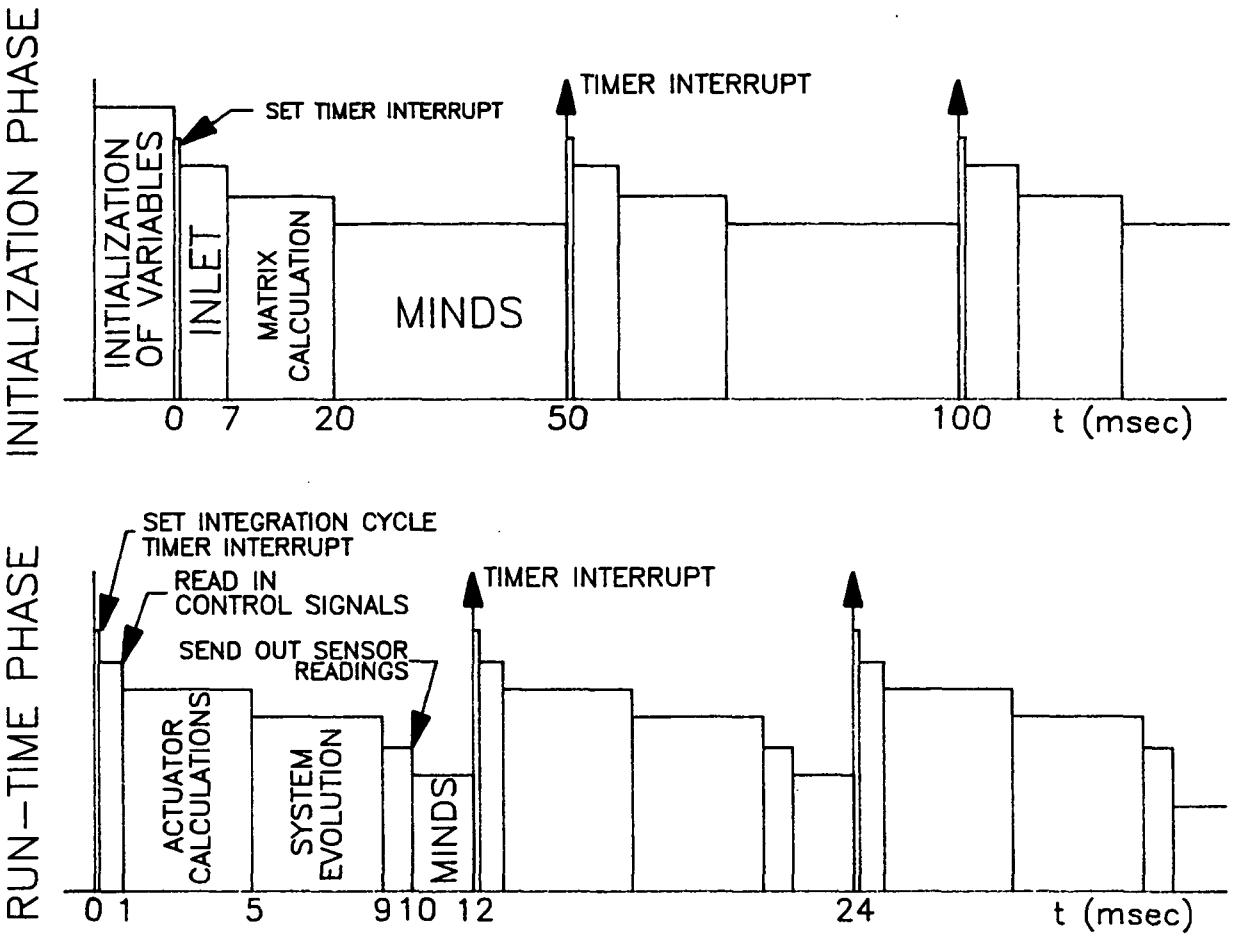

FIGURE 7. - TIMING DIAGRAMS.

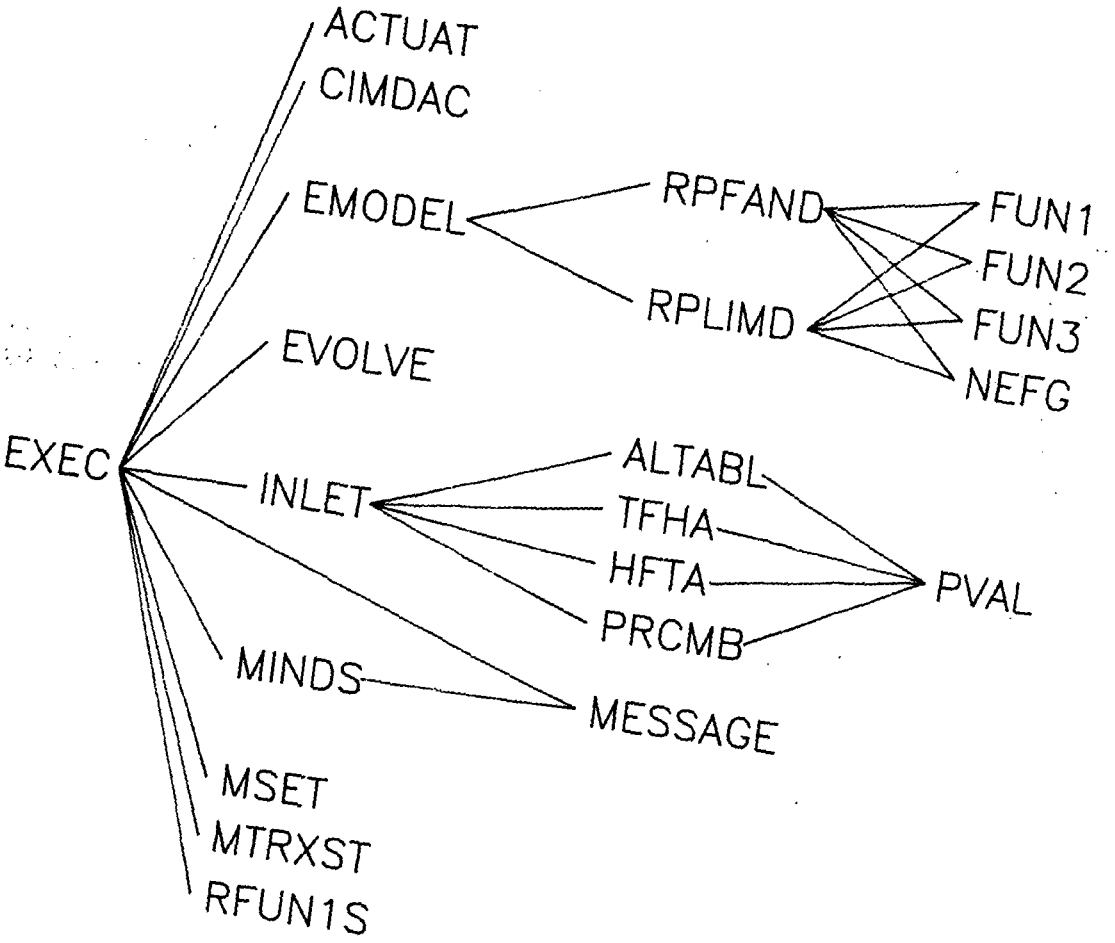

FIGURE 8. - HEIRARCHY OF SUBROUTINE CALLS.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-100889 AVSCOM TR-88-C-011		

4. Title and Subtitle

A Microprocessor-Based Real-Time Simulator of a Turbofan Engine

5. Report Date

6. Performing Organization Code

7. Author(s)

Jonathan S. Litt, John C. DeLaat, and Walter C. Merrill

8. Performing Organization Report No.

E-4124

9. Performing Organization Name and Address

NASA Lewis Research Center
Cleveland, Ohio 44135-3191
and
Propulsion Directorate
U.S. Army Aviation Research and Technology Activity—AVSCOM
Cleveland, Ohio 44135-3127

10. Work Unit No.

505-62-01

11. Contract or Grant No.

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, D.C. 20546-0001
and
U.S. Army Aviation Systems Command
St. Louis, Mo. 63120-1798

13. Type of Report and Period Covered

Technical Memorandum

14. Sponsoring Agency Code

15. Supplementary Notes

Prepared for the 19th Annual Pittsburgh Conference on Modeling and Simulation cosponsored by the ISA and IEEE, Pittsburgh, Pennsylvania, May 5-6, 1988.

16. Abstract

A real-time digital simulator of a Pratt and Whitney F100 engine is discussed. This self-contained unit can operate in an open-loop stand-alone mode or as part of a closed-loop control system. It can also be used in control system design and development. It accepts five analog control inputs and its sixteen outputs are returned as analog signals.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Microprocessor; Turbofan engine; Simulator; Real time; Sensors; Actuators	Unclassified – Unlimited Subject Category 07

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	16	A02

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161

National Aeronautics and
Space Administration

Lewis Research Center
Cleveland, Ohio 44135

NASA